Exploring Dual-Mode Training for Real-time Target Speaker Extraction

Li Li and Shogo Seki
AI Lab, CyberAgent, Japan
E-mail: {li_li, seki_shogo}@cyberagent.co.jp

***Abstract*—Implementing real-time target speaker extraction (TSE) presents significant challenges, primarily due to the need to minimize computational time to meet real-time constraints. Additionally, real-time TSE struggles to match the performance of offline systems because future information is inaccessible in streaming process. To enhance the performance of real-time models, this paper investigates the use of dual-mode training for real-time TSE. This approach involves training a single model with shared weights for both streaming and batch processing modes concurrently, with the goal of optimizing streaming processing performance through full-context modeling and joint training. The experiments are conducted on a TFGridNet-based TSE system, where we introduce a dual-mode implementation of TFGridNet. This implementation maintains the computational complexity of the streaming process while enabling dual-mode training. The experiments have indicated that dual-mode training is able to enhance the performance of streaming TSE system, particularly in terms of intelligibility metrics.**

I. Introduction

Target Speaker Extraction (TSE) [1], [2] is a pivotal technique in speech processing, designed to isolate the speech of a target speaker from observed audio mixtures. This process leverages various cues, including enrollment speech [3]–[5], spatial positioning [6], [7], and visual information [8]–[10], to accurately identify and extract the target speaker's speech. As speech interface applications, such as agent robots and smart speakers, become increasingly prevalent, the demand for real-time TSE systems is growing. Nonetheless, the development of high-performance, low-latency real-time TSE systems presents significant challenges. These challenges primarily arise from the necessity to optimize computational time on edge devices to meet stringent real-time processing requirements, specifically ensuring that the real-time factor (RTF) remains less than 1. Moreover, it is essential to implement TSE in a causal and streaming manner, implying that future information is inaccessible during processing.

One research direction involves optimizing model architecture to align with the hardware constraints of edge devices and achieve a suitable RTF from an algorithmic perspective [11]–[15]. For instance, VoiceFilter-Lite [11] achieves streaming processing and reduces model size by replacing the 8 convolutional layers in VoiceFilter [5] with 2 unidirectional long short-term memory (LSTM) layers and decreasing the number of fully connected (FC) layers. The End-to-End Enhancement Network (E3Net) [12] improves computational efficiency by employing a learnable encoder and decoder within a compact LSTM-based architecture, achieving a RTF of less than 0.1 with a $32\,\mathrm{ms}$ window size. LookOnceToHear [13] optimizes real-time processing by reducing the number of TFGridNet blocks [16] and kernel size for unfolding, removing group normalization, and limiting the time duration for looking back at past information, thereby realizing an end-to-end latency of less than $20\,\mathrm{ms}$.

Simply constraining architectures can limit the representation capability of neural networks, leading to performance degradation. To address this issue, several training techniques have been developed to mitigate such performance losses. One of the most widely used techniques is knowledge distillation (KD) [17], [18]. A common application of KD involves initially training a well-generalized teacher model and followed by training a student model using an additional KD loss alongside the standard loss to facilitate knowledge transfer between models. Typically, the teacher model with larger architectures employs batch or streaming processing, whereas the student model is a streaming model with smaller architectures. Although KD offers flexibility in selecting teachers with varying architectures and determining the type of knowledge to transfer, this flexibility also introduces challenges in making appropriate choices. The sequential training process requires more training time.

This paper explores an alternative training technique, dual-mode training [19], aimed at enhancing the performance of streaming models while maintaining computational efficiency during inference. Dual-mode training, a variant of multi-task learning, involves training a single model with shared weights to handle both streaming and batch processing tasks. This approach enables the streaming model to benefit from weight sharing and the joint training of full-context modeling provided by the batch processing model and obviates the necessity for two-stage training. Additionally, dual-mode models are anticipated to accommodate multiple scenarios for both streaming and batch processing using a single model, which significantly reduces storage requirements. Dual-mode training has been introduced to speech separation using a dual-path RNN [20] architecture [21], where the bidirectional LSTM is decomposed and reorganized so that the streaming model can share the weights of the bidirectional LSTM (BLSTM). However, the two implementations of dual-mode LSTM presented in [21] either increase the computational complexity of each layer in the streaming model by utilizing two LSTMs or reduce

979-8-3315-7206-8/25/$31.00

the shared knowledge by introducing separate FC layers for different modes.

In this paper, we propose a novel implementation for dual-mode LSTM and investigate a more intricate architecture characterized by reduced latency, specifically a causal TFGridNet with a latency of 20 ms. This configuration is selected due to TFGridNet's demonstrated state-of-the-art performance and extensive applicability in various speech enhancement and separation tasks. Moreover, a latency of 20 ms is more optimal for improving user experience. We utilize a multi-channel streaming TSE system based on the LookOnceToHear architecture [13], which employs both enrollment and spatial information as cues to adapt to complex environments with unspecified target speakers.

II. Method

A. Problem formulation

Consider a scenario where the target speaker is recorded by a microphone array in a noisy environment, including noise, reverberation, and interferences. The streaming TSE network $\mathcal{T}(\theta)$ extracts the target speech $\mathbf{s}_t = [s_{tI-I+1}, \ldots, s_{tI}]^\mathsf{T} \in \mathbb{R}^I$ based on the microphone array input $\mathbf{x}_t \in \mathbb{R}^{M\times I}$ and additional cues such as speaker embedding $\mathbf{e}$ and spatial information $\mathbf{r}_t$, which is expressed as

$$\hat{\mathbf{s}}_t = \mathcal{T}(\mathbf{x}_t, \mathbf{e}, \mathbf{r}_t; \theta). \tag{1}$$

Here, $\mathbf{x} = [x_1, \ldots, x_M]^\mathsf{T}$ denotes a mixture vector recorded by M microphones and t denotes the frame index, with each frame having a length of I. Note that the time-invariant speaker embedding $\mathbf{e}$ can be derived from pre-recorded enrollment speech when available. Alternatively, it can be computed from the initial multiple segments of noisy speech by utilizing spatial information to identify the target speaker

$$\mathbf{e} = \mathcal{E}(\mathbf{x}_{t=1:T_{\text{init}}}, \mathbf{r}_{t=1:T_{\text{init}}}). \tag{2}$$

Here, $\mathcal{E}$ represents a speaker embedding network that is trained by minimizing the cosine similarity between embeddings derived from noisy observations and their corresponding clean versions. T_{init} denotes a predefined number of frames for calculating the speaker embedding. This is similar to the approach used in LookOnceToHear [13], which assumes the target is positioned directly in front of the microphone array.

B. Causal TFGridNet for streaming TSE

TFGridNet employs a mapping-based approach, training a network to learn a function that maps the complex spectrogram of an observed signal to its clean version. The complex spectrogram $\mathbf{X} \in \mathbb{C}^{M\times F\times T}$ of a given observed mixture signal $\mathbf{x} \in \mathbb{R}^{M\times I\times T}$ is computed using the short-time Fourier transform (STFT), where F denotes the number of frequency bins. Convolutional 2D layers (Conv2D) are then utilized to embed each time-frequency bin, represented by a concatenated vector of its real and imaginary parts, into a D-dimensional latent representation $\mathbf{H}^0$. This representation is processed through multiple TFGridNet blocks, where intra-frame and inter-frame BLSTMs are used to separately model

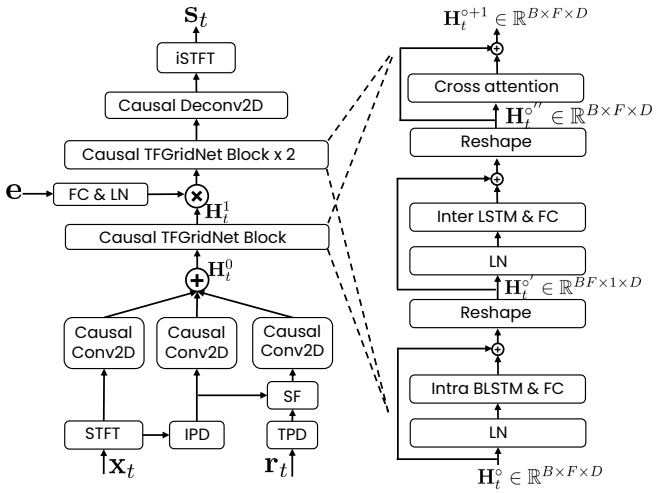

Fig. 1. Architecture of causal TFGridNet for streaming TSE. "B" denotes mini-batch size and other notations are defined in Subsec. II-B.

spectral and temporal relationships. Additionally, self-attention mechanisms are applied to effectively capture global temporal information. The clean signal is reconstructed by employing a deconvolution 2D layer (Deconv2D) to retrieve the complex spectrogram, followed by the inverse STFT (iSTFT) to convert it back into the waveform.

For streaming TSE, we adopt a causal version of TFGridNet [13], which takes $\mathbf{x}_t$ instead of $\mathbf{x}$ as input, as illustrated in Figure 1. To facilitate real-time streaming processing, several modifications are made to the original TFGridNet: the Conv2D and Deconv2D layers are replaced with their causal counterparts, group normalization is replaced by layer normalization (LN), and the inter-BLSTM used for capturing temporal information is substituted with a unidirectional LSTM. Furthermore, self-attention is replaced by cross-attention, which captures the relationship between the current frame and a limited number of past $N-1$ frames. To realize TSE, the speaker embedding $\mathbf{e}$ is fused with features $\mathbf{H}_t^1$ from the first causal TFGridNet block by mapping $\mathbf{e}$ to a D-dimensional vector via a fully connected layer (FC) and LN, followed by element-wise multiplication across frequencies. The fused features are then processed by two additional TFGridNet blocks.

The main difference from original TFGridNet lies in the feature extraction before the TFGridNet blocks (i.e., calculation of $\mathbf{H}_t^0$ in Figure 1). To leverage the information in multichannel signals, we use spectra $\mathbf{X}_t$ along with spatial features (SFs) derived from interchannel phase differences (IPD) and target-dependent phase differences (TPD) in 3D space [22]. These features are defined as:

$$\text{IPD}_{m,f,t} = \angle X_{1,f,t} - \angle X_{m,f,t} \quad (m = 2, \ldots, M), \tag{3}$$

$$\text{TPD}_{m,f,t} = \angle R_{1,f,t} - \angle R_{m,f,t} \quad (m = 2, \ldots, M), \tag{4}$$

$$\text{SF}_{f,t} = \sum_{2\le m\le M} \cos\left(\text{IPD}_{m,f,t} - \text{TPD}_{m,f,t}\right). \tag{5}$$

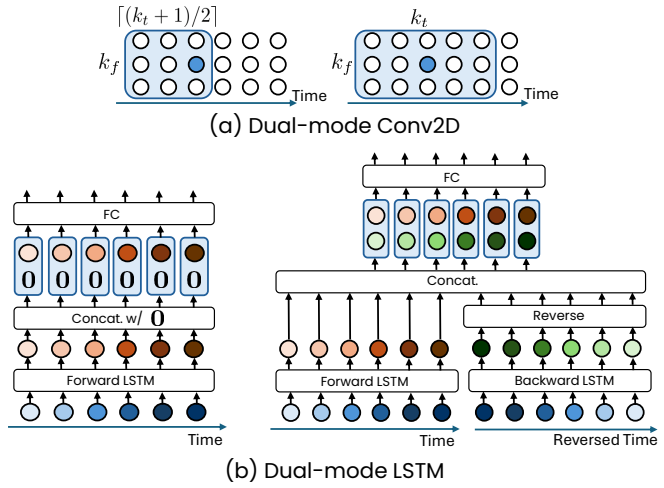

Fig. 2. Illustrations of dual-mode Conv2D layer (a) and dual-mode LSTM layer (b). Left and right sides in (a) demonstrate a causal Conv2D with kernel size of 3×3 and its non-causal version, respectively. Left and right sides in (b) demonstrate an LSTM and BLSTM, where $\mathbf{0}$ denotes a zero-vector.

Here, $\mathbf{R}_{f,t} = [R_{1,f,t}, \ldots, R_{M,f,t}]^\mathsf{T} \in \mathbb{C}^M$ denotes the steering vector calculated using the position information $\mathbf{r}_t$, and $f = 1, \ldots, F$ is the frequency index. $\mathbf{X}_t$, IPDs, and SFs are resepectively processed via a causal Conv2D and then added to form a comprehensive feature $\mathbf{H}_t^0$.

C. Dual-mode implementation of TFGridNet

To harness the benefits of weight sharing with batch processing models and facilitate knowledge transfer through full-context modeling, it is crucial to integrate the streaming model and batch processing model into a unified network. Two design principles are considered for dual-mode models in the previous work [19]: (P1) each layer in a dual-mode model should operate in either streaming or batch processing mode, and (P2) the design of the dual-mode layer should not significantly increase the number of additional parameters compared to the streaming model. We adhere to these principles in designing the dual-mode TFGridNet, although we will experimentally demonstrate that (P2) is not strictly necessary. Specifically, we adopt dual-mode convolutional layers, LSTM, and attention mechanisms, while keeping other layers normal, as they are frame-wise operations.

Dual-mode convolutional layers, including Conv2D, are implemented by constructing common symmetric convolution kernels and utilizing only the portions corresponding to the current and past frames in streaming mode. For example, for a symmetric Conv2D kernel with a size of $k_f \times k_t$ used in batch processing, only the kernel of size $k_f \times \lceil (k_t+1)/2 \rceil$, constructed using a binary mask, is employed for streaming processing. An illustration of the dual-mode Conv2D is shown in Figure 2(a).

Two implementations of dual-mode LSTM have been proposed, referred to as "decomposition" and "reorganization," respectively [21]. However, the "decomposition" approach introduces separate FC layers for LSTM and BLSTM, which may limit the amount of knowledge that can be shared. In contrast, the "reorganization" approach forwards input signals to both forward and backward LSTMs in both streaming and batch processing modes, thereby increasing the computational complexity of the streaming mode. To address these limitations, we propose a novel dual-mode LSTM implementation, as illustrated in Figure 2(b). Specifically, we utilize a single FC layer for both streaming and batch modes. To complement the mismatch in output feature size, we concatenate zero vectors, corresponding to the backward LSTM output, with the forward LSTM output. This implementation maximizes the shared knowledge between modes while maintaining the computational complexity of the streaming mode unchanged.

The implementation of the dual-mode attention mechanism is achieved by controlling the target frames used to compute the attention vector. Given the feature $\mathbf{H}_t^{\circ''}$ output by the inter-LSTM, where $^\circ$ denotes any TFGridNet block, the key $\mathbf{K}_t^l \in \mathbb{R}^{1\times(F\times E)}$, query $\mathbf{Q}_t^l \in \mathbb{R}^{1\times(F\times E)}$, and value $\mathbf{V}_t^l \in \mathbb{R}^{1\times(F\times D/L)}$ are calculated by passing $\mathbf{H}_t^{\circ''}$ through a FC layer, followed by PReLU and LN. Here, E denotes the dimension of the key and query, and $l = 1, \ldots, L$ denotes the index of heads. The output of l-th attention head $\mathbf{A}_t^l \in \mathbb{R}^{1\times(F\times D/L)}$ is calculated using different sets of keys and values. For batch processing, keys obtained from all frames are used, represented as $\mathbf{K}_T^l = [\mathbf{K}_1^{l\,\mathsf{T}}, \ldots, \mathbf{K}_T^{l\,\mathsf{T}}]^\mathsf{T} \in \mathbb{R}^{T\times(F\times E)}$. For streaming processing, keys obtained from the current and past $N-1$ frames are utilized, denoted as $\mathbf{K}_N^l = [\mathbf{K}_{t-N+1}^{l\,\mathsf{T}}, \ldots, \mathbf{K}_t^{l\,\mathsf{T}}]^\mathsf{T} \in \mathbb{R}^{N\times(F\times E)}$. Similarly, $\mathbf{V}_T^l \in \mathbb{R}^{T\times(F\times D/L)}$ and $\mathbf{V}_N^l \in \mathbb{R}^{N\times(F\times D/L)}$ are used for batch and streaming processing, respectively. The attention output is given by

$$\mathbf{A}_t^l = \mathrm{softmax}\left(\frac{\mathbf{Q}_t^l \mathbf{K}_u^{l\,\mathsf{T}}}{\sqrt{F\times E}}\right)\mathbf{V}_u^l, \tag{6}$$

where $u \in \{T, N\}$ denotes using either the situation indicated by the subscript. The final output is obtained by concatenating the outputs of L heads.

D. Loss

We employ a weighted sum negative signal-to-noise ratio (SNR) loss to balance the streaming and batch processing modes, which is expressed as

$$\mathcal{L} = \alpha\mathcal{L}_S + \mathcal{L}_B \tag{7}$$

$$\mathcal{L}_{v\in\{S,B\}} = -\mathbb{E}[\mathrm{SNR}(\hat{\mathbf{s}}_\mathrm{t}, \mathbf{s}_\mathrm{t})]. \tag{8}$$

Here, α represents the relative significance of streaming mode compared to batch processing mode, denoted by S and B respectively.

III. Experimental Evaluations

A. Datasets

We extracted speech utterances from LibriSpeech [23] and noise samples from WHAM! [24] to generate noisy mixtures with two or three speakers, resulting in 100,000, 5,000, and

TABLE I

MEAN VALUES AND 95% CI OF PESQ, STOI, SDR, SI-SNR, AND LSD FOR **STREAMING PROCESSING**. "S" AND "B" PRESENT THE NUMBER OF BLOCKS USED IN STREAMING AND BATCH MODE, RESPECTIVELY. "T" INDICATES THE NUMBER OF BLOCKS IN THE TEACHER MODEL. * DENOTES STATISTICAL SIGNIFICANCE AGAINST THE BASELINE STREAMING MODEL: $^*p < 0.05$, $^{**}p < 0.001$. THE BEST SCORE FOR EACH METRIC IS SHOWN IN BOLD.

Method	#Blocks	α	PESQ ↑	STOI [%] ↑	SDR [dB] ↑	SI-SNR [dB] ↑	LSD [dB] ↓
Noisy	—	—	1.14±0.00	36.59±0.25	-3.09±0.09	-3.26±0.09	4.62±0.05
Streaming	$S3$	—	1.59±0.01	55.78±0.29	6.42±0.07	5.28±0.08	3.95±0.04
KD_output	$S3T3$	—	1.62±0.01 **	56.52±0.28 **	6.62±0.07 **	5.48±0.08 **	3.89±0.04 **
	$S3T6$	—	1.63±0.01 **	56.82±0.28 **	6.55±0.07 **	5.46±0.08 **	3.89±0.04 **
KD_feature	$S3T3$	—	**1.64±0.01** **	57.53±0.28 **	6.47±0.07 **	5.54±0.08 **	3.83±0.04 **
	$S3T6$	—	**1.64±0.01** **	57.34±0.28 **	6.47±0.07 **	5.50±0.08 **	3.82±0.04 **
Dual-mode	$S3B3$	1	1.60±0.01 **	57.01±0.28 **	6.15±0.07 **	5.20±0.08 **	3.86±0.04 **
		2	1.61±0.01 **	56.77±0.28 **	6.39±0.07 *	5.34±0.08 **	3.84±0.04 **
		3	1.62±0.01 **	57.17±0.28 **	6.49±0.07 **	5.43±0.08 **	3.88±0.04 **
		4	1.63±0.01 **	57.32±0.28 **	6.58±0.07 **	5.48±0.08 **	**3.79±0.04** **
		5	1.61±0.01 **	56.80±0.28 **	6.50±0.07 **	5.45±0.08 **	3.88±0.04 **
	$S3B6$	1	1.59±0.01	56.72±0.28 **	6.31±0.07 **	5.29±0.08	3.91±0.04 **
		2	**1.64±0.01** **	57.60±0.28 **	6.64±0.07 **	5.61±0.08 **	3.86±0.04 **
		3	1.62±0.01 **	**57.76±0.28** **	6.59±0.07 **	5.61±0.08 **	3.87±0.04 **
		4	1.63±0.01 **	57.60±0.28 **	6.67±0.07 **	5.62±0.08 **	3.88±0.04 **
		5	1.63±0.01 **	57.62±0.28 **	**6.69±0.07** **	**5.65±0.08** **	3.89±0.04 **

TABLE II

MEAN VALUES AND 95% CI OF PESQ, STOI, SDR, SI-SNR, AND LSD FOR **BATCH PROCESSING**. ASYMMETRIC CIS ARE FORMATTED AS "MEAN [LOWER, UPPER]". "S" AND "B" PRESENT THE NUMBER OF BLOCKS USED IN STREAMING AND BATCH MODE, RESPECTIVELY. * DENOTES STATISTICAL SIGNIFICANCE AGAINST THE BASELINE BATCH MODEL HAVING THE SAME NUMBER OF BLOCKS: $^*p < 0.05$, $^{**}p < 0.001$. THE BEST SCORE FOR EACH METRIC IS SHOWN IN BOLD.

Method	#Blocks	α	PESQ ↑	STOI [%] ↑	SDR [dB] ↑	SI-SNR [dB] ↑	LSD [dB] ↓
Noisy	—	—	1.14±0.00	36.59±0.25	-3.09±0.09	-3.26±0.09	4.62±0.05
Batch	$B3$	—	1.82±0.01	61.32±0.28	8.09±0.07	6.98±0.07	3.83±0.04
	$B6$	—	1.94±0.01	64.90 [64.64, 65.17]	**8.84±0.07**	**7.75±0.07**	3.81±0.04
Dual-mode	$S3B3$	1	1.80±0.01 **	61.35±0.27	7.63±0.07 **	6.73±0.07 **	3.80±0.04 **
		2	1.76±0.01 **	60.14±0.28 **	7.53±0.07 **	6.53±0.07 **	3.82±0.04 **
		3	1.73±0.01 **	59.41±0.28 **	7.41±0.07 **	6.35±0.07 **	3.93±0.04 **
		4	1.76±0.01 **	60.05±0.28 **	7.52±0.07 **	6.44±0.07 **	3.81±0.04 **
		5	1.71±0.01 **	58.93±0.28 **	7.27±0.07 **	6.28±0.07 **	3.94±0.04 **
	$S3B6$	1	**1.97±0.01** **	65.33 [65.07, 65.60] **	8.63±0.07 **	7.70±0.07 **	**3.75±0.04** **
		2	1.95±0.01 **	65.24 [64.98, 65.51] **	8.62±0.07 **	7.71±0.07 *	3.78±0.04 **
		3	1.94±0.01	**65.34±0.26** **	8.66±0.06 **	7.71±0.07 *	3.77±0.04 **
		4	1.93±0.01 **	65.06±0.26 **	8.56±0.07 **	7.63±0.07 **	3.76±0.04 **
		5	1.93±0.01 **	64.70±0.27 **	8.43±0.07 **	7.54±0.07 **	3.78±0.04 **

10,000 utterances for training, validation, and test sets, respectively. Using the image method in Pyroomacoustics [25], we generated 28,000 room impulse responses for linear microphone arrays with $M = 7$ elements, split into 20,000, 5,000, and 3,000 for training, validation, and testing. Room sizes ranged from [4, 3, 2] to [8, 6, 3] m, with reverberation times (T60) from 0.15 to $0.80\,\mathrm{s}$. The microphone spacing was fixed at $0.028\,\mathrm{m}$, and speakers were positioned within an angular range of $\pm 60^\circ$ in front of the array. We used dynamic mixing during training. The validation and test datasets included 5,000 and 10,000 samples, respectively. All signals were resampled to $16\,\mathrm{kHz}$ and fixed at $5\,\mathrm{s}$ length. Noisy enrollment utterances were generated similarly, which were considered as the first utterance spoken by the target speaker in real-world applications.

B. Experimental settings

We used TFGridNet [16] with parameters[1] $D = 64$, $B = 3$, $H = 64$, $I = 1$, $J = 1$, $L = 4$, and $E = 8$ for the network. The number of frames N used in the streaming mode was set to 50. The STFT window and hop sizes were set to $12/8\,\mathrm{ms}$. This resulted in a network with an algorithmic delay of $12\,\mathrm{ms}$ and a real-time factor of 0.67 on an Intel Xeon Platinum 8452Y CPU. Speaker embeddings were derived from noisy enrollment data by utilizing positional information to identify the target speaker. This process employed a noisy embedding network, which was trained to minimize the cosine similarity between its output embeddings and those generated by Resemblyzer[2], an open-source speaker encoder used as the clean speaker embedding model. More network details are available in [13]. We trained all networks using Adam optimizer with a learning rate (lr) of 5.0×10^{-4}, halved when the validation loss did not improve for five consecutive epochs, down to a minimum lr of 1.0×10^{-6}. The batch size was 12, and training ran for 50 epochs. We explored two dual-mode model configurations, differing in the number of TFGridNet blocks within the batch processing mode. The

[1]Note that notations here follow the original TFGridNet [16].

[2]https://github.com/resemble-ai/Resemblyzer

first configuration, termed $S3B3$, utilized 3 blocks, identical to the streaming mode. The second configuration, termed $S3B6$, employed 6 blocks, with the streaming mode using the 2nd, 4th, and 6th blocks and skipping the other three blocks. For comparison, we trained streaming models using KD with pre-trained batch models serving as the teacher. The terms $S3T3$ and $S3T6$ denote the use of teacher models with 3 blocks and 6 blocks, respectively. We distilled the output knowledge, referred to as "KD_output," by applying a negative SNR loss between the outputs of the teacher and student models. Additionally, we distilled feature knowledge, referred to as "KD_feature," by utilizing an $L1$ loss between the features generated by the last TFGridNet block in both the teacher and student models. The weight assigned to the KD loss was set at 1.0.

We used perceptual evaluation of speech quality (PESQ) [26], short-time objective intelligibility (STOI) [27], signal-to-distortion ratio (SDR) [28], scale-invariant SNR (SI-SNR) [29], and log-spectral distance (LSD) as evaluation metrics.

C. Results and discussion

Table I presents the mean values and corresponding 95% confidence intervals (CIs) obtained in streaming mode. A paired samples t-test was conducted to assess the statistical significance of the results between the dual-mode models and their respective baselines. The significance levels are denoted as follows: $p \geq 0.05$ with no mark indicates no statistical significance, $^{*}p < 0.05$ indicates statistical significance, and $^{**}p < 0.001$ indicates high statistical significance. We found all dual-mode models demonstrated improvements in PESQ, STOI, and LSD metrics compared to the baseline streaming model. Meanwhile, there was a degradation in SDR and SI-SNR when $\alpha = 1$ was applied across both configurations, and when $\alpha = 2$ was used in the $S3B3$ configuration. These findings suggest that leveraging knowledge from full-context modeling may be more beneficial for enhancing perceptual quality than for improving signal-level quality. When comparing the results of the two configurations, $S3B6$ demonstrated further improvements in SDR and SI-SNR while maintaining the same levels in other metrics as $S3B3$. This suggests that a larger batch processing model, which can learn more comprehensive features and enhance capabilities, is more beneficial for improving the streaming model. This finding also challenges the second principle (P2) proposed in previous work [19], indicating that a batch model with an increased number of parameters can be effectively utilized and may offer greater benefits when paired with an appropriate setting. The results achieved by models trained with KD demonstrate the differences in effectiveness with different KD options. Distilling output knowledge achieved better SDR and SI-SNR, while distilling feature knowledge resulted in more improvements in other metrics. In contrast, the dual-mode model $S3B6$, except for $\alpha = 1$, achieved better or comparable performance. Especially, $\alpha = 2$ surpassed the best scores achieved by different KD settings in terms of all metrics.

Table II presents the results for the batch processing mode. In the $S3B3$ configuration, we observed that nearly all metrics experienced degradation, with the exception of LSD, which showed improvement when α was set at 1, 2, and 5. Conversely, the degradation in the $S3B6$ configuration was less pronounced, and dual-mode training even enhanced performance metrics such as PESQ, STOI, and LSD. When combining the results from both modes, we found that increasing α can enhance SDR and SI-SNR in streaming mode, albeit at the expense of other metrics. Setting α at 2 or 3 in the $S3B6$ configuration strikes the best balance, achieving reasonable improvements in streaming mode while maintaining comparable performance in batch processing mode. This suggests that knowledge from the large batch model should be utilized as a form of regularization rather than treating both modes equivalently.

We also explored initializing the dual-mode model with a pre-trained batch model and applying KD loss related to the output or latent features, as described in [19], [21]. However, these techniques did not result in any further performance improvements in our experiments.

IV. Conclusions

This paper investigates dual-mode training for real-time TSE, specifically focusing on a TFGridNet-based TSE network with a system latency of $20\,\mathrm{ms}$. We introduced a novel implementation of dual-mode LSTM that maximizes shared weights while keeping the computational complexity of the streaming mode unchanged. Experimental results demonstrated that dual-mode training is beneficial for enhancing the performance of the streaming model. Moreover, with an appropriate training configuration, it is possible to develop a unified model capable of performing both streaming and batch processing, achieving even better performance than separate streaming and batch processing models.

References

[1] K. Zmolikova, M. Delcroix, T. Ochiai, K. Kinoshita, J. Černocký, and D. Yu, "Neural Target Speech Extraction: An overview," *IEEE Signal Processing Magazine*, vol. 40, no. 3, pp. 8–29, 2023. DOI: 10.1109/MSP.2023.3240008.

[2] T. Ochiai, M. Delcroix, T. Moriya, *et al.*, "Target sound information extraction: Speech and audio processing with neural networks conditioned on target clues," *Acoustical Science and Technology*, vol. 46, no. 3, pp. 197–209, 2025.

[3] M. Delcroix, K. Zmolikova, T. Ochiai, K. Kinoshita, and T. Nakatani, "Speaker Activity Driven Neural Speech Extraction," in *Proc. ICASSP*, 2021, pp. 6099–6103.

[4] M. Ge, C. Xu, L. Wang, E. S. Chng, J. Dang, and H. Li, "SpEx+: A Complete Time Domain Speaker Extraction Network," in *Proc. INTERSPEECH*, 2020, pp. 1406–1410.

[5] H. R. Muckenhirn, I. L. Moreno, J. Hershey, *et al.*, "VoiceFilter: Targeted Voice Separation by Speaker-Conditioned Spectrogram Masking," in *Proc. INTERSPEECH*, 2019, pp. 2728–2732.

[6] R. Gu, L. Chen, S.-X. Zhang, *et al.*, "Neural Spatial Filter: Target Speaker Speech Separation Assisted with Directional Information," in *Proc. INTERSPEECH*, 2019, pp. 4290–4294.

[7] J. Lin, P. Wang, H. Dinkel, *et al.*, "Focus the Sound around You: Monaural Target Speaker Extraction via Distance and Speaker Information," in *Proc. INTERSPEECH*, 2023, pp. 2488–2492.

[8] D. Michelsanti, Z.-H. Tan, S.-X. Zhang, *et al.*, "An overview of deep-learning-based audio-visual speech enhancement and separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1368–1396, 2021.

[9] T. Ochiai, M. Delcroix, K. Kinoshita, A. Ogawa, and T. Nakatani, "Multimodal speakerbeam: Single channel target speech extraction with audio-visual speaker clues.," in *INTERSPEECH*, 2019, pp. 2718–2722.

[10] R. Tao, X. Qian, Y. Jiang, J. Li, J. Wang, and H. Li, "Audio-visual target speaker extraction with selective auditory attention," *IEEE Transactions on Audio, Speech and Language Processing*, 2025.

[11] Q. Wang, I. L. Moreno, M. Saglam, *et al.*, "Voicefilter-lite: Streaming targeted voice separation for on-device speech recognition," *arXiv preprint arXiv:2009.04323*, 2020.

[12] M. Thakker, S. E. Eskimez, T. Yoshioka, and H. Wang, "Fast real-time personalized speech enhancement: End-to-end enhancement network (e3net) and knowledge distillation," in *Interspeech 2022*, 2022, pp. 991–995. DOI: 10.21437/Interspeech.2022-10962.

[13] B. Veluri, M. Itani, T. Chen, T. Yoshioka, and S. Gollakota, "Look Once to Hear: Target Speech Hearing with Noisy Examples," in *Proc. CHI*, 2024, pp. 1–16.

[14] H. Sato, T. Moriya, M. Mimura, *et al.*, "Speakerbeam-ss: Real-time target speaker extraction with lightweight conv-tasnet and state space modeling," in *Interspeech 2024*, 2024, pp. 5033–5037. DOI: 10.21437/Interspeech.2024-413.

[15] S. E. Eskimez, T. Yoshioka, H. Wang, X. Wang, Z. Chen, and X. Huang, "Personalized speech enhancement: New models and comprehensive evaluation," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Ieee, 2022, pp. 356–360.

[16] Z.-Q. Wang, S. Cornell, S. Choi, Y. Lee, B.-Y. Kim, and S. Watanabe, "TF-GridNet: Integrating Full- and Sub-Band Modeling for Speech Separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 3221–3236, 2023.

[17] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015. [Online]. Available: http://arxiv.org/abs/1503.02531.

[18] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[19] J. Yu, W. Han, A. Gulati, *et al.*, "Dual-mode asr: Unify and improve streaming asr with full-context modeling," in *ICLR 2021*, ICLR 2021, 2021. [Online]. Available: https://arxiv.org/abs/2010.06030.

[20] Y. Luo, Z. Chen, and T. Yoshioka, "Dual-path rnn: Efficient long sequence modeling for time-domain single-channel speech separation," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 46–50.

[21] K. Li and Y. Luo, "On the design and training strategies for rnn-based online neural speech separation systems," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2023, pp. 1–5.

[22] Y. Shao, S.-X. Zhang, and D. Yu, "Multi-Channel Multi-Speaker ASR Using 3D Spatial Feature," in *Proc. ICASSP*, 2022, pp. 6067–6071.

[23] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. ICASSP*, 2015, pp. 5206–5210.

[24] G. Wichern, J. Antognini, M. Flynn, *et al.*, "WHAM!: Extending Speech Separation to Noisy Environments," in *Proc. INTERSPEECH*, 2019, pp. 1368–1372.

[25] R. Scheibler, E. Bezzam, and I. Dokmanić, "Pyroomacoustics: A python package for audio room simulation and array processing algorithms," in *Proc. ICASSP*, 2018, pp. 351–355.

[26] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs," in *Proc. ICASSP (Cat. No. 01CH37221)*, vol. 2, 2001, pp. 749–752.

[27] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "A short-time objective intelligibility measure for time-frequency weighted noisy speech," in *Proc. ICASSP*, 2010, pp. 4214–4217.

[28] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[29] J. Le Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "SDR–half-baked or well done?" In *Proc. ICASSP*, 2019, pp. 626–630.